EXHIBIT 21

MUNG CHIANG

Networked Life

20 Questions and Answers

CAMBRIDGE UNIVERSITY PRESS
Cambridge, New York, Melbourne, Madrid, Cape Town
Singapore, São Paulo, Delhi, Mexico City

Cambridge University Press
The Edinburgh Building, Cambridge CB2 8RU, UK

Published in the United States of America by Cambridge University Press, New York

www.cambridge.org
Information on this title: www.cambridge.org/9781107024946

First published 2012

Printed in the United States by Edwards Brothers

A catalog record for this publication is available from the British Library

ISBN 978-1-107-02494-6 Hardback

Additional resources for this publication at www.cambridge.org/networkedlife

Contents

17 IPTV and Netflix: How can the Internet support video?

We saw in Chapter 13 that the Internet provides a "best effort," i.e., "no effort" service. So, how can it support video distribution that often imposes stringent demands on throughput and delay?

17.1 A Short Answer

17.1.1 Viewing models

Watching video is a significant part of many people's daily life, and it is increasingly dependent on the Internet and wireless networks. Movies, TV shows, and home videos flow from the cloud through the IP network to mobile devices. This trend is changing both the networking and the entertainment industries. As of 2011, there were more than 100 million IPTV users in the USA, and Youtube and Netflix together take up about half of the Internet capacity usage. As the trend of decoupling among contents, content delivery channels, and content-consuming devices intensifies, IP has become the basis of almost all the content distribution systems.

This trend is bringing about a revolution in our viewing habits.

- *Content type*: Both user-generated and licensed content have become prevalent. Clearly, more user-generated content implies an increasing need for upload capacity, which is traditionally designed to be much smaller than download capacity.
- *When*: For many types of video content, we can watch them anytime we want, with the help of devices like a Digital Video Recorder (DVR) on IPTV or services like HBO Go.
- *Where*: We can watch video content almost anywhere, at least anywhere with a sufficiently fast Internet connection.
- *How*: Instead of just on the TV and desktop computers, we can watch video on our phones, tablets, and any device with a networking interface and a reasonable screen.
- *How much*: We are watching more video, thanks to applications like Netflix, Hulu, Deja, and embedded videos on many websites. For example, in February 2012, Hulu had roughly 31 million unique viewers, watching 951 million

videos. Comcast NBS Universal had 39 million unique viewers, watching 205 million videos (more than doubling the number from summer 2011). Some of these are free, some are free but with intrusive advertisements, some require a monthly subscription, some are pay-per-view, and some are part of a bundled service (e.g., the triple play of IPTV, Internet access, and VoIP). If the Internet connection charge is usage-based, there is also the "byte-transportation" cost per GB, as discussed in Chapter 11.

We can categorize viewing models along four dimensions. Each combination presents different implications to the network design in support of the specific viewing model.

- *Real time vs. precoded*: Some videos are watched as they are generated in real time, e.g., sports, news, weather videos. However, the vast majority are precoded: the content is already encoded and stored somewhere. In some cases, each video is stored with hundreds of different versions, each with a different playback format or bit rate. Real-time videos are more sensitive to delay, while precoded videos have more room to be properly prepared. Some other video-based services are not only real-time, but also two-way interactive, e.g., video calls, video conferencing, and online gaming. Clearly, interactive video has even more stringent requirements on delay and jitter (i.e., the variance of delay over time).
- *Streaming or download*: Some videos, like those on Netflix and YouTube, are streamed to you, meaning that your device does not keep a local copy of the video file (although Netflix movies sometimes can be stored in a local cache, and YouTube has started a movie-rental service). In other cases, e.g., iTunes, the entire video is downloaded first before played back at some later point. Of course, the content itself may be automatically erased from local storage if digital rights are properly managed, like in movie rentals. In-between these two modes there is the possibility of *partial* download and playback. As shown in the Advanced Material, this reduces the chance of jitter, and is followed in practice almost all the time except for interactive or extremely real-time content.
- *Channelized or on-demand*: Some contents are organized into channels, and you have to follow the schedule of each channel accordingly. This is the typical TV experience we have had for decades. Even with DVR, you still cannot jump the schedule in real time. In contrast, Video on Demand (VoD) allows you to get the content when you want it. Both YouTube and Netflix are VoD. There are also VoD services on TV, usually charging a premium. Sometimes the content owner changes the model, e.g., in 2011 HBO in the USA changed to a VoD model with its HBO Go services on computers and mobile devices. In-between the two extremes, there is *NVoD*, Near Video on Demand, which staggers the same channel every few minutes, so that, within a latency tolerance of that few minutes, you get the experience of VoD.

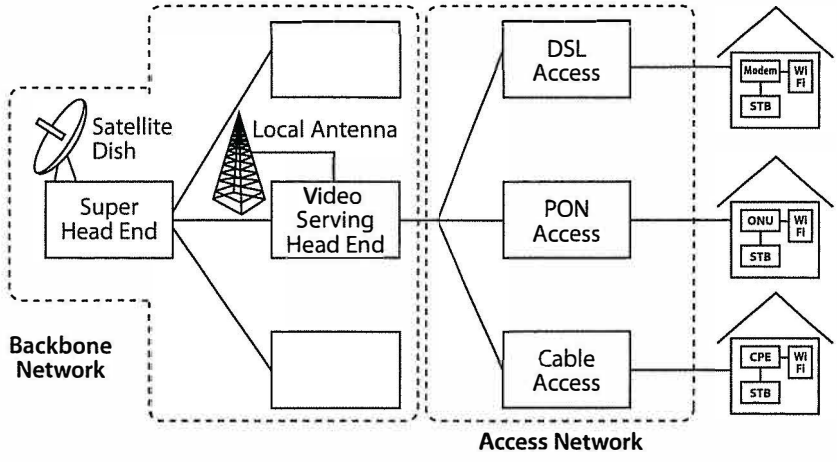


Figure 17.1 A typical architecture of IPTV. The content is collected at the super head end and distributed to different local video-serving head ends across the country, which also collect local content. Then it is further distributed to access networks running on copper, fiber, or cable, before reaching the homes. This is often carried out in private networks owned and managed by a single ISP.

- *Unicast or multicast*: Unicast means transmission from one source to one destination. Multicast means from one source to many destinations, possibly tens of millions for events like the Olympic Games, that belong to a multicast group. An extreme form of multicast is *broadcast*: everyone is in the multicast group. If you do not want certain content, you do not have to watch it, but it is sent to you anyway. TV is traditionally multicast, sometimes through physical media that are intrinsically multicast too, such as satellite. The Internet is traditionally unicast. Now the two ends are getting closer. We see unicast capabilities in IP-based video distribution, but also multicast in IP networks (carried out either in the network layer through IP multicast routing or in the application layer through P2P).

It seems that there are $2^4 = 16$ combinations using the above taxonomy of video viewing modes. Obviously some combinations do not make sense, for example, real-time video must be streaming-based and cannot be download-based. But precoded video can be either streaming- or download-based. Or, true VoD cannot be multicast since each individual asks for the content at different times, but channelized or NVoD content can be either unicast or multicast.

17.1.2 IP video: IPTV and VoI

The term "IP video" actually encompasses two styles: (1) IPTV and (2) VoI. IPTV turns TV channel delivery into IP-based, whereas VoI views the Internet as a pipe that can simply support any type of content delivery. Increasingly VoI

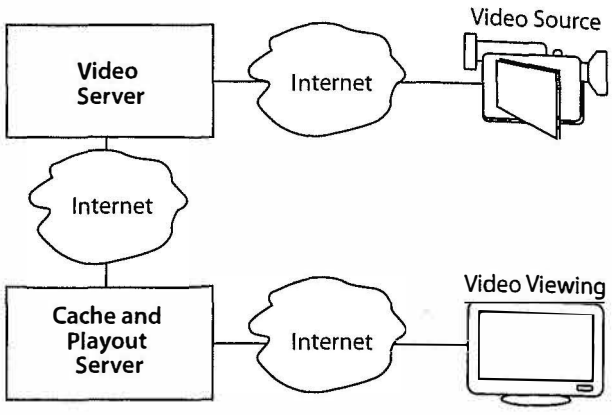


Figure 17.2 A typical architecture of video over the Internet. Video sources, ranging from iPhones to professional video cameras, upload content to video servers, which then distribute them through local caches to the viewers around the world who download the video to their devices. This is all carried out in the public Internet.

is becoming the more popular way for people to consume video content than IPTV. If all content is available on demand through VoI, what advantages does IPTV offer to the consumer experience?

(1) **IPTV** is often included as part of the triple- or quadruple-play service bundle provided by an ISP. It is delivered over a *private and managed network*, with a set-top box on the customer's premises. This private network uses IP as a control protocol, but many parts of it are deployed and operated by a single ISP, e.g., a telephone or cable company offering the Internet access. This makes it easier to control the quality of service. The content is often channelized, multicast, and streaming-based but with recording capability using DVR.

Before TV turned to the Internet access networks, it was delivered primarily through one of the following three modes: broadcast over the air, via satellites, or through cables. So why is the IPTV revolution happening now? There are a few key reasons.

- *Convergence*: almost everything else is converging on IP, including phone calls. Putting video on IP makes it a unified platform to manage.
- *Cost*: Having a uniform platform reduces the costs of maintaining separate networks.
- *Flexibility*: IP has demonstrated that one of its greatest strengths is the ability to support diverse applications arising in the future.
- *Compression* has got better and access network *capacity* has increased sufficiently that it has become possible to send HD TV channels.

(2) **Video over the Internet** (**VoI**) is delivered entirely over *public networks*, often via unicast, and to a variety of consumer devices. Given the current evolution of business models, VoI is increasingly taking over the IPTV business

as consumers access video content over the IP pipes without subscribing to TV services. There are three main types of VoI.

- The content owner sends videos through server–client architectures without a fee, e.g., YouTube, ABC, and the BBC.
- The content owner sends videos through server–client architectures with a fee, e.g., Netflix, Amazon Prime, Hulu Plus, and HBO Go.
- Free P2P sharing of movies, e.g., Bit Torrent and PPLive.

We touched upon the revenue models for IPTV and VoI. As to the cost models, they often consist of the following items.

- *Content*: The purchase of content-distribution rights. Popular and recent movies and TV series are naturally more expensive.
- *Servers*: The installation and maintenance of storage and computing devices.
- *Network capacity*: The deployment or rental of networking capacity to move content around and eventually deliver it to consumers.
- *Customer premises equipment*, such as set-top boxes and games consoles.
- *Software*: The software systems that manage all of the above and interface with consumers.

Whether it is IPTV or VoI, the quality measures depend on the bit rate, delay, and jitter. What kind of bit rates do we need for videos? It depends on a few factors, e.g., the amount of motion in the video, the efficiency of the compression methods, the screen resolution, and the ratio of viewing distance and screen size. But, generally speaking, the minimum requirement today is about 300 kbps. Below that, the visual quality is just too poor even on small screens. For standard-definition movies we need at least 1 Mbps, and a typical movie takes $1-2$ GB. For high-definition movies we need at least $6-8$ Mbps, and a typical movie takes $5-8$ GB. Truly HD video needs $20-25$ Mbps to be delivered. And the latest standard of UltraHD needs over 100 Mbps. As we will see in the next section, to make IP video work, we need technologies from both multimedia signal processing and communication networking.

17.2 A Long Answer

As shown in Figure 17.3, the overall protocol stack for IP video includes the following: MPEG over HTTP/SIP/IGMP/RTSP, over RTP/UDP/TCP, over IP, over ATM or Ethernet or WiFi, over wireless/fiber/DSL/cable. In this section, we go into some detail regarding the top three layers: compression, application, and transport, trying to highlight interesting networking principles beyond an "alphabet soup" of acronyms.

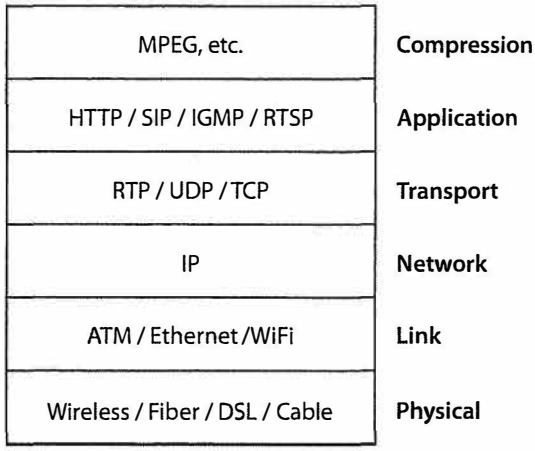


Figure 17.3 A layered network architecture in support of video traffic. Compression standards such as MPEG use various application layer protocols, which in turn rely on combinations of transport and network layer protocols. The focus of this section is on a few key ideas in the top three layers.

17.2.1 Compression

A video is a sequence of frames moving at a particular speed. Each frame is a still picture consisting of **pixels**. Each pixel is described by its colors and luminance digitally encoded in bits. The number of bits per frame times the number of frames per second gives us the **bit rate** of a video file. Typical frame rates are 25 or 29.97 frames per second for standard definition, and 50 or 60 frames per second for high definition. Typical pixels per frame for high-definition video are $1280 \times 720 = 921600$ or $1920 \times 1080 = 20736004$. If we had to send all these bits, we would not have been able to deliver video of any reasonable quality to the vast majority of Internet-connected devices today.

In order to put more content into a given pipe, we need **compression**. This is the process of taking out redundancies in signals. If the resulting file can be later recovered, say at the consumer device, to be exactly the same as the original one, it is called **lossless compression**, e.g., the Lempel–Ziv compression used in zipping files. Otherwise, it is called **lossy compression**, and there is a tradeoff between the compression ratio (the size of the file after compression relative to that before compression) and the resulting fidelity. This is called the **rate–distortion** tradeoff, as shown in Figure 17.4.

In many VoI services, each video clip is precoded into many bitstreams, each at a different point on the rate–distortion curve. Then a specific one is chosen to be sent over the Internet, depending on any combination of the following three factors:

- the screen resolution of the end-user device,
- the throughput supported by the end-to-end path between the video server and the device, and

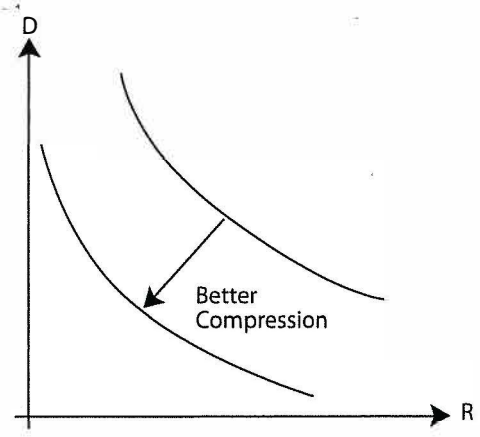


Figure 17.4 Rate–distortion curves for two different lossy compression schemes. Distortion can be measured by objective metrics or subjective tests. A higher rate leads to lower distortion. The closer to the origin the tradeoff curve, the better the tradeoff.

- as in the recent proposal of Quota-Aware-Video-Adaptation (QAVA), the data quota usage pattern of each user under usage-pricing plans in Chapters 11.

There are many techniques to help achieve the best shape and the largest range of such a tradeoff by taking out redundancies in the bits, e.g., transform coding (seek structures in the frequency domain of the signals), Huffman coding (reduce the expected length of the compressed signal by making frequently appearing codes shorter, as we saw in Chapter 10), and perceptual coding for video (let people's perceptual process guide the choice of which pixels and which frames to compress). Many frames also look alike. After all, that is how a perception of continuous motion can be registered in our brain. So video compressors often only need to keep track of the differences between the frames, leading to a significant saving in the number of bits needed to represent a group of pictures.

Video compression has made substantial progress since the early 1990s.

- **MPEG1**, a standard in 1992: this was used for VCD (which uses 1 Mbps bit rate).
- **MPEG2** (called H.262 by a different standardization body called the ITU-T), a standard in 1996: this was used for DVD (which uses about 10 Mbps bit rate).
- **MP3**, the layer 3 of the MPEG2 standard (there is no MPEG3, that track of standard was absorbed into MPEG2): this popular standard for the online music industry is for encoding just audio, and can achieve a 12:1 compression ratio.
- **MPEG4**, a standard in 2000: this is the current video compression standard family.

- MPEG4 Part 10 (also called AVC or H.264) in 2004: with 16 profiles, it offers substantial flexibility. It is also at least twice as good as MPEG2's compression capability. It is used for HDTV (with 20 – 25 Mbps bit rate) and Blu-ray (with 40 Mbps bit rate). It can readily achieve a compression factor of 100.
- There are also other non-MPEG formats: H.261 was popular in IP video in the early days, and Quick Time by Apple is merging into MPEG4. There are also Windows Media Player by Microsoft, Flash by Adobe, and Real Media Viewer by Real Networks.

A key idea in MPEG compression is to exploit the redundancy across frames when the motion is not rich. This is called motion compensation with inter-frame prediction. There are three types of frames, and collectively a set of them form a **Group of Pictures** (GoP).

- **I frame (Intra-coded)**: This is an independent frame. Its encoding does not depend on the frames before or after it.
- **P frame (Predictive-coded)**: This type of frame depends on the previous I (or P) frame, but not the one after it.
- **B frame (Bidirectionally predictive-coded)**: This type of frame depends on both the I (or P) frames before and after it.

Each GoP must start with an I frame, followed by a sequence of P and B frames, as shown in Figure 17.5. The I frame is the most important one, while P and B frame losses are much more tolerable. At the same time, I frames are

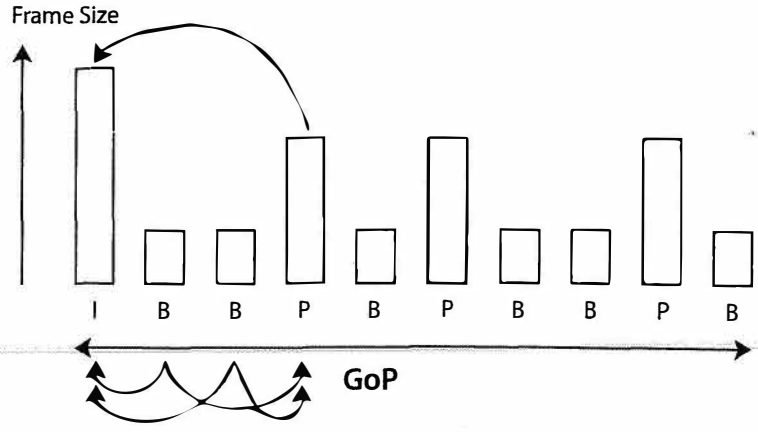


Figure 17.5 A typical structure of a Group of Pictures (GoP). Each GoP starts with an I frame, followed by a sequence of B and P frames. The I frame is independent and the most important one in each GoP. Each P frame depends on the previous I/P frame. Each B frame depends on both the previous and the following I/P frames. Some of these dependence relationships are indicated by arrows. Choosing the length and structure of a GoP affects bit rates, error resiliency, and delay in channel change.

harder to compress than P and B frames, which use motion prediction to assist in compression.

The length of a GoP influences several metrics.

- *Bitrate efficiency*: If the GoP is longer, there are more P and B frames (the more easily compressible ones). The bit rate becomes lower.
- *Error resilience*: If an I frame is lost and, consequently, the entire GoP needs to be retransmitted, a longer GoP means that more frames need to be retransmitted. There is a tradeoff between efficiency and resilience, as we will quantify in a homework problem.
- *Instant channel change*: For channelized video content, the ability to change channels fast is important if the traditional TV viewing experience is to be replicated on IPTV. Since GoP represents the logical basic unit for playback, a longer GoP means that the viewer needs to wait longer to change channels. There are also other factors at play for channel change, such as multicast group operations to be explained next.

17.2.2 Application layer

In addition to the ability to do multicast routing, we also need Internet Group Management Protocol (**IGMP**). It runs the multicast group management and tells the router that a client (an end-user device) wants to join a particular multicast group. There are only two essential message types: the router asks a `membership-query` to clients, and each client replies with a `membership-report` telling the router which groups it belongs to. There is an optional message `leave-group` from clients to routers. This message is optional because, by the principle of *soft state* (explained in Chapter 19), if the membership report information does not periodically refresh the groups, the client is assumed to leave the group.

Joining and leaving multicast groups incur propagation and processing delays. Instant channel change may become frozen. To accelerate a channel change, an IPTV service provider may send some unicast GoP to the end-user when she first requests a channel change. It goes into multicast mode once the "join group" request is processed. There is clearly a tradeoff between network efficiency and user experience. We will see more tradeoffs like this involved in optimizing the networked delivery of video.

For streaming applications, we often use Real Time Streaming Protocol (**RTSP**). It allows a media player to control the transmission of a media stream, e.g., fast forward, rewind, pause, play. It is independent of the compression standard or transport protocol.

A typical procedure is shown in Figure 17.6. The request for content first runs over HTTP from a web browser to a web server. Then, knowing the type of media and compression used (by reading the response received from the web server), the client can open the right media player, which then uses RTSP to carry out

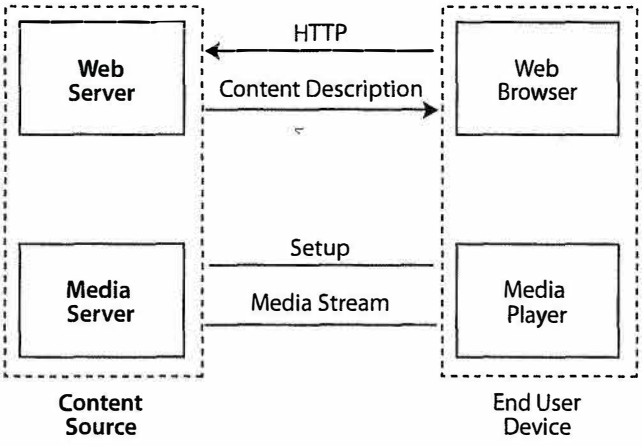


Figure 17.6 Real Time Streaming Protocol (RTSP) at work. An HTTP session first enables control information to be exchanged between the client web browser and the web server. This is followed by the actual media stream from the media server to the media-player software on the end-user device.

message passing between the client and the media server that actually holds the video file. Unlike HTTP, RTSP must keep track of the current state of the file at the client media player so that it can operate functionalities like pause and play. You must have realized there is a lot of overhead associated with managing video traffic, and we will see more of such overhead in Chapter 19.

Another protocol often used for IP multimedia transmission is Session Initiation Protocol (**SIP**) from the Internet Engineering Task Force (IETF). It establishes a call between a caller and callee over an IP network, determines the IP address, and manages the call, e.g., adds callers, transfers or holds calls, or changes voice encoding. Together with video standards, SIP can also provide a mechanism for video conferencing.

Yet another commonly used protocol is **H.323** from the International Telecommunication Union (ITU). It is actually a large suite of protocols involving many components already discussed in this subsection.

As you can see, there are many standardization bodies: the IETF standardizes many Internet related protocols, the ITU and the Institute of Electrical and Electronic Engineers (IEEE) have many standardization bodies, and some major standards have their own governing bodies, e.g., 3GPP and 3GPP2 for cellular, WiMax Forum for WiMax, DSL Forum for DSL, MPEG for video compression, etc. The way they operate is a mix of technology, business, and political factors, but they all share a common goal of inter-operability among devices so that the positive network effect of technology adoption can be achieved.

17.2.3 Transport layer

We have seen the *connection-oriented* transport protocol of TCP in Chapter 14. But much multimedia traffic, especially real-time or interactive types, runs

instead over User Datagram Protocol (**UDP**) in the transport layer. UDP is *connectionless*, meaning that it does not try to maintain end-to-end reliability, not even sort packets in the right order. It works with IP to deliver packets to the destination, but if the packets do not get there, it will not try to solve that problem. For example, Skype uses UDP unless the client sits behind a firewall that allows only TCP flows to pass through. UDP also handles multicast well, since it does not require the receipt of a packet at all the destinations. IGMP that we just saw often runs on UDP.

UDP is fundamentally different from TCP in the end-to-end control of the Internet. Why would applications with tight deadlines prefer UDP? It boils down to the tradeoff between timeliness and reliability.

- TCP uses a three-way handshake (explained in Chapter 19) to establish a session, whereas UDP does not introduce that latency. TCP uses congestion control to regulate source rates, whereas UDP sends out the packet as soon as it is generated by the application layer.
- TCP ensures reliability through packet retransmission in the transportation layer, but many multimedia applications have their own built-in error resilience in the application layer. For example, losing a B frame in a GoP can often be concealed in a media player so that the viewers cannot tell. Moreover, a lost and retransmitted packet will likely be too late to be useful in playback by the time it arrives at the destination. It is instead more important to avoid holding back playback and just proceed.

In addition to real-time or interactive multimedia, many network management or signaling protocols, like SNMP in Chapter 19 and RIP in Chapter 13, also run on top of UDP. For these signaling protocols, there are two more reasons to prefer UDP.

- TCP maintains too many states for each session compared with UDP. So UDP can support many more parallel sessions at the same time.
- TCP header is 20 bytes and UDP is 8 bytes. For small control packets, this difference in overhead matters.

A protocol on top of UDP is Real-time Transport Protocol (**RTP**), which is heavily used in many IP multimedia applications including VoIP. It specifies a format to carry multimedia streams. The key challenge here is to support many types of media formats, including new ones coming up. And the key solution is to specify a range of profiles and payload formats, specific to each media type, without making the header dependent on the media type. RTP runs on the data plane that transmits the actual data, and its companion RTP Control Protocol (**RTCP**) runs on the control plane that sends control signals. RTCP keeps track of the RTP stream's information, such as the number of packets, the number of bytes, and the timestamp information. It monitors the statistics and synchronizes multiple streams.

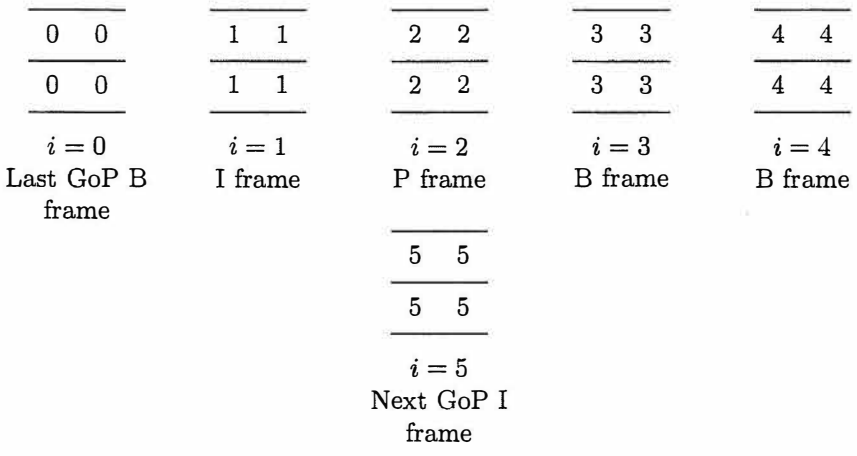
Now that we have finished an overview of the top three layers in Figure 17.3, we will proceed to two examples of tradeoffs in IP video delivery.

17.3 Examples

17.3.1 Video quality and I/P/B frames

In this example, we will explore the effect of dropped I, P, and B frames on video quality. Consider each grayscale value of the pixel at position (x, y) in frame i. The transmitted frame is $\bar{p}_i$ and the received frame is p_i. For our metric of video quality, we use the L-1 norm, the sum of absolute differences across all the pixels and frames, i.e., error $= \sum_{x,y,i} |p_i(x, y) - \bar{p}_i(x, y)|$.

Consider a very small example with 2×2 pixels. For simplicity, assume all pixel values in a given frame are the same (this is a very boring video). A GoP consists of four frames, indexed by $i = 1, 2, 3, 4$. We will drop each frame of the GoP in turn, and quantify the effect on our error metric. Assume that frame 0 and frame 5 (belonging to the preceding and following GoPs, respectively) are always correctly received.



Recall that an I frame has no reference frame, a P frame uses the last I or P frame as the reference frame, and a B frame uses the last I or P frame and the next I or P frame as reference frames. An "error" in frame i means that either frame i is missing or it has to perform error concealment because frame i's reference is missing. We can set up a few error-concealment rules at the receiver.

- If the receiver misses any frame, it instead displays the last available frame.
- If the receiver detects an error in the reference frame of a P frame, it also displays the last available frame in place of the P frame.
- If the receiver detects an error in a reference frame of a B frame, it displays the other reference frame.

If the I frame of this GoP is dropped, the receiver displays what is summarized in Table 17.1.

	$\bar{p}(1,1)$	$\bar{p}(1,2)$	$\bar{p}(2,1)$	$\bar{p}(2,2)$
$i=0$	0	0	0	0
$i=1$	0	0	0	0
$i=2$	0	0	0	0
$i=3$	5	5	5	5
$i=4$	5	5	5	5
$i=5$	5	5	5	5

Table 17.1 Frame 1: Dropped, so repeat frame 0. Frame 2: Error in reference frame 1, so repeat frame 1. Frame 3: Error in reference frame 2, so display frame 5. Frame 4: Error in reference frame 2, so display frame 5.

Then the error between the transmitted and the received picture is

$$
\begin{aligned}
&|p_1(1,1)-\bar{p}_1(1,1)|+|p_1(1,2)-\bar{p}_1(1,2)|+|p_1(2,1)-\bar{p}_1(2,1)|+|p_1(2,2)-\bar{p}_1(2,2)|\\
&+|p_2(1,1)-\bar{p}_2(1,1)|+|p_2(1,2)-\bar{p}_2(1,2)|+|p_2(2,1)-\bar{p}_2(2,1)|+|p_2(2,2)-\bar{p}_2(2,2)|\\
&+|p_3(1,1)-\bar{p}_3(1,1)|+|p_3(1,2)-\bar{p}_3(1,2)|+|p_3(2,1)-\bar{p}_3(2,1)|+|p_3(2,2)-\bar{p}_3(2,2)|\\
&+|p_4(1,1)-\bar{p}_4(1,1)|+|p_4(1,2)-\bar{p}_4(1,2)|+|p_4(2,1)-\bar{p}_4(2,1)|+|p_4(2,2)-\bar{p}_4(2,2)|\\
&=4|p_1(1,1)-\bar{p}_1(1,1)|+4|p_2(1,1)-\bar{p}_2(1,1)|\\
&+4|p_3(1,1)-\bar{p}_3(1,1)|+4|p_4(1,1)-\bar{p}_4(1,1)|\\
&=4|1-0|+4|2-0|+4|3-5|+4|4-5|\\
&=24.
\end{aligned}
$$

If instead the P frame is dropped, the receiver displays what is summarized in Table 17.2.

The error between the transmitted and the received picture is

$$
\begin{aligned}
&4|p_1(1,1)-\bar{p}_1(1,1)|+4|p_2(1,1)-\bar{p}_2(1,1)|\\
&+4|p_3(1,1)-\bar{p}_3(1,1)|+4|p_4(1,1)-\bar{p}_4(1,1)|\\
&=4|1-1|+4|2-1|+4|3-5|+4|4-5|\\
&=16.
\end{aligned}
$$

If the first B frame, frame 3, is dropped, the receiver displays what is summarized in Table 17.3.

	$\bar{p}(1,1)$	$\bar{p}(1,2)$	$\bar{p}(2,1)$	$\bar{p}(2,2)$
$i=0$	0	0	0	0
$i=1$	1	1	1	1
$i=2$	1	1	1	1
$i=3$	5	5	5	5
$i=4$	5	5	5	5
$i=5$	5	5	5	5

Table 17.2 Frame 2: Dropped, so repeat frame 1. Frame 3: Error in reference frame 2, so display frame 5. Frame 4: Error in reference frame 2, so display frame 5.

	$\bar{p}(1,1)$	$\bar{p}(1,2)$	$\bar{p}(2,1)$	$\bar{p}(2,2)$
$i=0$	0	0	0	0
$i=1$	1	1	1	1
$i=2$	2	2	2	2
$i=3$	2	2	2	2
$i=4$	4	4	4	4
$i=5$	5	5	5	5

Table 17.3 Frame 3: Dropped, so repeat frame 2.

The error between the transmitted and the received picture is

$$\begin{aligned}
&4|p_1(1,1)-\bar{p}_1(1,1)|+4|p_2(1,1)-\bar{p}_2(1,1)|\\
&+4|p_3(1,1)-\bar{p}_3(1,1)|+4|p_4(1,1)-\bar{p}_4(1,1)|\\
&=4|1-1|+4|2-2|+4|3-2|+4|4-4|\\
&=4.
\end{aligned}$$

If the second B frame, frame 4, is dropped, the receiver displays what is summarized in Table 17.4.

The error between the transmitted and the received picture is

$$\begin{aligned}
&4|p_1(1,1)-\bar{p}_1(1,1)|+4|p_2(1,1)-\bar{p}_2(1,1)|\\
&+4|p_3(1,1)-\bar{p}_3(1,1)|+4|p_4(1,1)-\bar{p}_4(1,1)|\\
&=4|1-1|+4|2-2|+4|3-3|+4|4-3|\\
&=4.
\end{aligned}$$

The greatest error resulted from dropping the I frame, followed by the P frame, and finally the B frames. More important frames cause more error in the GoP when dropped, leading to a bigger drop in the visual quality.

	$\bar{p}(1,1)$	$\bar{p}(1,2)$	$\bar{p}(2,1)$	$\bar{p}(2,2)$
$i=0$	0	0	0	0
$i=1$	1	1	1	1
$i=2$	2	2	2	2
$i=3$	3	3	3	3
$i=4$	3	3	3	3
$i=5$	5	5	5	5

Table 17.4 Frame 4: Dropped, so repeat frame 3.

17.3.2 Latency–jitter tradeoff

In this example, we look at the effect of initial buffering on streaming video's playback. We will see what the playback latency should be to provide a smooth viewing experience.

Assume there is one frame per packet. Figure 17.7 shows the transmitted, arrived (at the receiver), and played frames over time. We refer to these as the timing curves, abbreviated as the V, A, and P curves. The V and A curves are given by the source and the network, and our job is to design the best P curve.

The video source transmits at a constant rate, so V is a superposition of unit step functions. We abuse the notation a little to use vectors to represent the discrete jumps on the curves. Let V_i denote the time at which packet i is transmitted, A_i the time at which packet i is received, and P_i the time at which packet i is displayed to the user. The delay between the transmission and receipt of packet i is given by $d_i = A_i - V_i$. We have

$$\mathbf{V} = \begin{bmatrix} 1 \\ 2 \\ 3 \\ 4 \end{bmatrix}, \qquad \mathbf{A} = \begin{bmatrix} 5.8 \\ 7.5 \\ 8 \\ 9.4 \end{bmatrix}, \qquad \mathbf{d} = \mathbf{A} - \mathbf{V} = \begin{bmatrix} 4.8 \\ 5.5 \\ 5 \\ 5.4 \end{bmatrix}.$$

In reality, we cannot know $\mathbf{A}$ ahead of time and have to either estimate it or adapt in real time, like in a homework problem. For now, we assume it is known. P must be a unit step function since frames need to be displayed at a constant rate. When should the playback begin, i.e., what is the value of P_1? The goal is to minimize the total delay experienced by the user:

$$\begin{aligned} \text{minimize} \quad & \sum_i (P_i - V_i) \\ \text{subject to} \quad & P_{i+1} = P_i + 1, \ \forall i \\ & P_i \geq A_i, \ \forall i \\ \text{variables} \quad & \{P_i\}. \end{aligned}$$

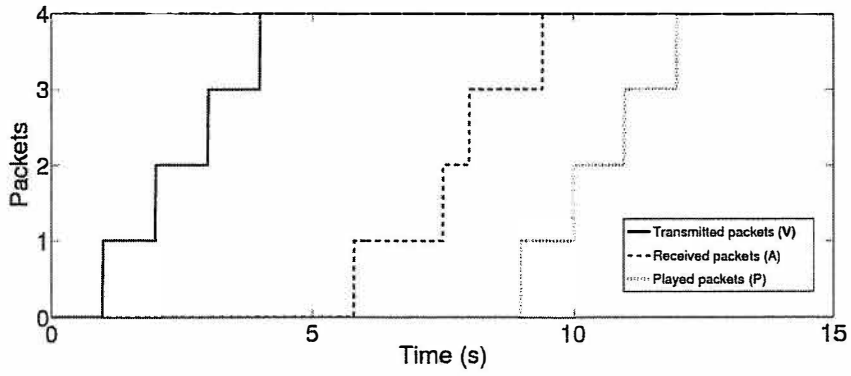


Figure 17.7 Playback buffer smoothes video playback. This graph shows curve V at the source: how packets are transmitted with a constant rate; curve A at the receiver: how the packets' arrival times vary as they traverse the network, and curve P at playback: how playback latency can smooth the arrival jitter. The V and A curves are given, and the P curve needs to be designed to make sure it is a superposition of unit step functions, lies below the A curve, and yet is as far to the left as possible.

The first constraint in the above optimization ensures that the playback curve P is a unit step function, and the second constraint says that playback of a packet can occur only after the packet has been received. The objective function can be further simplified; since P and V are both unit step curves, $P_i - V_i$ is the same for all i, so the objective function is equivalent to minimizing any single $P_i - V_i$.

This problem can be solved easily through visual inspection. Essentially, we shift a unit step function P to the left, until $P_k = A_k$ for some k, and $P_i \geq A_i \ \forall i \neq k$. That is, we want to make curve P as close to curve A as possible but still remain below A.

From Figure 17.7, clearly $k = 2$. Since $P_2^* = A_2 = 7.5$ s, we have $P_1^* = 6.5$ s. This means playback should begin at 6.5 s, which in turn means delaying playback by $P_1 - A_1 = 6.5 - 5.8 = 0.7$ s. This playback buffering latency avoids frozen video due to the variation of packet arrivals through the network.

In general, for a constant-rate source, the playback should begin at $P_1^* = A_1 + D$, where $D = \max_i(d_i - d_1)$ and represents the maximum delay variation. This formula applies to the above example. Since $D = d_2 - d_1 = 5.5 - 4.8 = 0.7$ s, we have $P_1^* = A_1 + D = 5.8 + 0.7 = 6.5$ s.

17.4 Advanced Material

There are three general approaches in managing the quality of service on top of the simple connectivity services, which is offered by the thin waist of TCP/IP in the best effort style.

- Treat different sessions differently during resource allocation.

- Regulate which clients can be admitted.
- Distribute servers to strategic locations.

As an analogy, think of a highway's traffic control. Differentiating resource allocation is like reserving a lane for certain vehicles, like a car-pool lane. Regulating admission is like using the on-ramp traffic lights during the rush hours. Distributing servers is like constructing new exits for popular destinations such as grocery stores, which is clearly a much-longer-timescale operation than the other two.

17.4.1 Queueing policies

Different sessions can be treated differently, inside a node or along a link in a network. For example, there are several standard queueing disciplines in a router. As shown in Figure 17.8, the following three methods will create different sequences of service timing among the incoming sessions. But they are all work-conserving: they do not waste a timeslot if there is some packet to be served.

- *Round robin*: Each class of traffic takes turns.
- *Weighted fair queueing*: While taking turns, one class can receive a higher rate than another.
- *Priority queueing*: Higher-priority-class packets are processed before lower-priority ones, which have to wait until there are no more higher-priority packets in the queue.

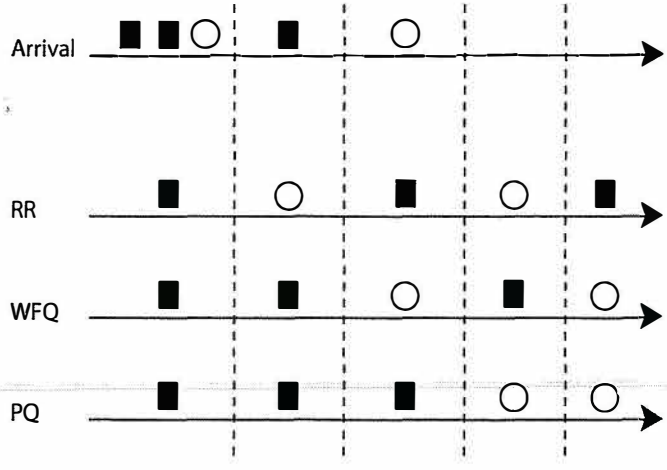


Figure 17.8 Three queueing disciplines give different orders of packet service. There are two classes arriving over the timeslots denoted by dotted lines. In Round Robin (RR) scheduling, the square packets and circle packets are served in turn, one packet in each timeslot. In Weighted Fair Queuing (WFQ), the two classes take turns, but square packets get a higher service rate. In Priority Queuing (PQ), square packets have strict priority, and circle packets get their chance only when all square packets have been sent.

What constitutes a *fair* allocation among competing sessions? This is yet another instance where we touch upon the notion of fairness. We will see a systematic treatment of the subject in Chapter 20.

Of course, differential treatment methods as above do not provide a guarantee on the quality of service. For that, we need methods for resource *reservation*. There are dynamic versions of establishing end-to-end circuits in the network layer and reserving adequate resources (such as capacity, timeslot, and processing power) so that the end-user experience is guaranteed to be good enough. Handoff in mobile networks in Chapter 19 will offer such an example.

17.4.2 Admission control

An alternative approach to managing resource competition is to regulate the demand. In some sense, TCP congestion control does that in a feedback loop on the timescale of RTT. Policing or throttling further shapes the rate of traffic injection into the network. Time-dependent pricing in Chapter 12 is another form of admission control on a longer timescale.

For an *open-loop* control, we can use admission control: deciding whether a session should be allowed to start transmitting at any given timeslot. Such admission control is usually carried out at the edge of the network, often at the first network element facing the end-user devices.

One possible admission control is to control the peak rate of traffic injection over some timescale. This is like the ramp light that regulates cars getting onto a highway and smoothes the traffic injection into the highway network during the rush hours. By reducing the rate of green lights at the ramp, we can lower the rate of adding traffic onto the highway (the backbone network) at the expense of causing congestion at the ramp (the access or edge network).

An example of admission control is **leaky bucket**, shown in Figure 17.9. In order for each packet to be admitted, there must be a token dripping from a (conceptual) leaky bucket. The bucket drips tokens at a rate of a, and has a volume of B tokens. In a homework problem, we will see that weighted fair queueing and leaky bucket can together shape any arrival patterns to a desirable one.

17.4.3 Content distribution

Yet another approach, in addition to differentiating resource allocation and controlling user admission, is to change the location of the source of content, so that the content is brought closer to the users, as shown in Figure 17.10. This leverages the continuous drop of storage cost and the widespread popularity of certain content to create *spatially pipelined* distribution of content.

The idea of changing the location of the source has long been practiced in web proxy servers since the mid-1990s. ISPs cache popular web content at local storage closer to end-users. A whole industry sector has also been generated,

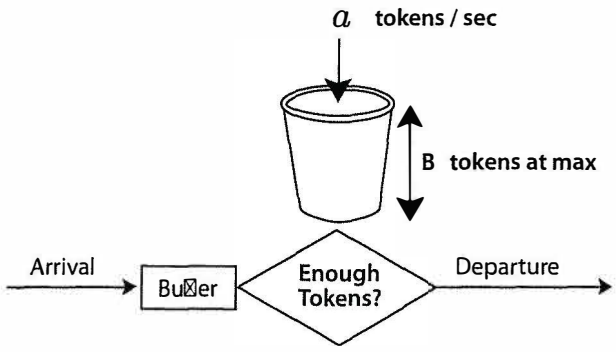


Figure 17.9 A leaky bucket for admission control. Each bucket can hold at most B tokens, and r tokens are added to the bucket per second. Each transmitted packet consumes one token. In order for a packet in the buffer to be transmitted on the egress link, there must be a token available in the bucket.

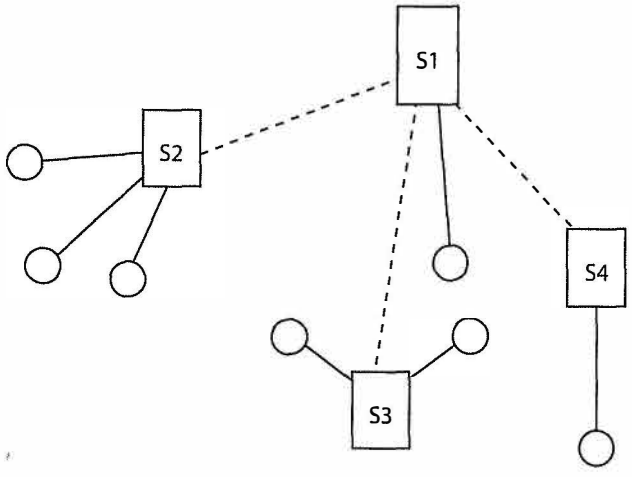


Figure 17.10 An illustration of a content distribution networks. The content originally placed in server S1 is replicated in three other locations, S2, S3, and S4, being sent through fiber links shown by dotted lines. When a client requests a piece of content, a particular server is selected as the source to serve that demand.

operators of **Content Distribution Networks** (CDN). They serve either ISPs or content owners, and manage the following processes.

- Deploy many servers, sometimes in private server farms and sometimes in shared data centers. These are often called mirror sites.
- Replicate content and place it in different servers. This involves optimization on the basis of the availability of high-speed links and high-capacity storage, as well as prediction of content popularity in different geographic areas.
- For each content request, select the right server to serve as the content source. This server-selection optimization tries to minimize the delay perceived

by a user. It is run by the CDN operator, under a given routing decided by the ISP. On the other hand, ISP runs routing optimization through traffic engineering like in Chapter 13, under a given traffic demand pattern between servers and user-devices. These two optimization problems are "mirror images" of each other, and the corresponding operations by the CDN operator and the ISP may have unintended interactions.

The convergence of content owner and content transporters is creating interesting dynamics. A proper operation of CDNs can create a win–win situation: consumers get better quality of service (as delay is reduced and throughput increased), while content owners or ISPs reduce the cost of deploying large-capacity servers and pipes (as the congestion in the network is reduced).

Summary

Box 17 Quality-of-service mechanisms

Contents are being decoupled from specific content delivery channels and content-consuming devices, and IP has become the basis of almost all the content distribution systems, including IPTV and VoI. Video compression, video network applications, and connectionless transport protocols all contribute to the proliferation of video on the Internet. Quality differentiation, admission control, and content distribution intelligence enable the best-effort Internet to support video applications.

Further Reading

The subject matter of this chapter spans both analytic models of quality of service and the systems design of multimedia protocols.

1. The fundamentals of video signal processing can be found in many graduate textbooks on the subject, including the following recent one:

A. C. Bovik, *The Essential Guide to Video Processing*, Academic Press, 2009.

2. The following book provides a concise summary of all the major video-over-IP systems, including IPTV and VoI:

W. Simpson, *Video over IP*, 2nd edn., Focal Press, 2008.

3. The following standard textbook of computer networking provides much more details about the application and transport layer protocols we mentioned, like IGMP, RTSP, SIP, UDP, and RTP:

J. Kurose and K. Ross, *Computer Networking: A Top-Down Approach*, 5th edn., Addison Wesley, 2009.

4. The following classic paper combines leaky-bucket admission control and generalized processor sharing to provide a delay guarantee:

A. Parekh and R. G. Gallager, "A generalized processor sharing approach to flow control in integrated services networks: The single-node case," *IEEE/ACM Transactions on Networking*, vol. 1, no. 3, pp. 344–357, June 1993.

5. The following book provides a concise survey both of the deterministic, algebraic approach and of the stochastic, "effective-bandwidth" approach to the design of quality guarantees in a network:

C. S. Chang, *Performance Guarantees in Communication Networks*, Springer Verlag, 2000.

Problems

17.1 *Video-viewing models* ⋆

Fill in the table indicating which video-watching models are infeasible. Provide examples of companies that follow each feasible model. Some rows have been filled out as an example.

Real-time or precoded	Streaming or download	Channelized or on-demand	Unicast or multicast	Companies
Real-time	Streaming	Channelized	Unicast	
Real-time	Streaming	Channelized	Multicast	
Real-time	Streaming	On-demand	Unicast	
Real-time	Streaming	On-demand	Multicast	
Real-time	Download	Channelized	Unicast	
Real-time	Download	Channelized	Multicast	
Real-time	Download	On-demand	Unicast	
Real-time	Download	On-demand	Multicast	
Precoded	Streaming	Channelized	Unicast	
Precoded	Streaming	Channelized	Multicast	

Precoded	Streaming	On-demand	Unicast	YouTube, Hulu, NBC, HBO Go
Precoded	Streaming	On-demand	Multicast	Infeasible (on-demand multicast)
Precoded	Download	Channelized	Unicast	
Precoded	Download	Channelized	Multicast	
Precoded	Download	On-demand	Unicast	
Precoded	Download	On-demand	Multicast	

17.2 *Compression–reliability tradeoff* ⋆

Let us examine the tradeoff between compression and error resilience through a back-of-the-envelope calculation. Suppose we have 15 frames to transmit and two possible GoP structures: (1) IPB and (2) IPBBB. Suppose an I frame costs 7 kB, a P frame costs 3 kB, and a B frame costs 1 kB.

If an entire GoP is not received correctly, we assume that the GoP must be sent again. As our metric of error resilience, consider the expected number of bits that must be retransmitted at least once. The probability of dropping a frame is 1% and assumed to be independent. (These assumptions are made to simplify this homework problem. In a realistic setting, if a P or B frame in a GoP is lost, the entire GoP does not need to be retransmitted. Loss is not independent and usually much less than 1%. And there should be many more frames in a video clip.)

(a) In case 1, the video frame structure is IPB/IPB/IPB/IPB/IPB. What is the total cost of the video in kB? What is the cost per GoP per kB?

(b) What is the probability that an entire GoP is transmitted successfully in case 1? What is the expected number of GoPs that are successful on the first attempt at the transmission of the entire video? What is the expected number of GoPs that must be retransmitted at least once? How much does the first retransmission cost in kB?

(c) Repeat (a) for case 2, where the video frame structure is now: IPBBB/IPBBB/IPBBB.

(d) Repeat (b) for case 2.

(e) Compare you results from (a), (b), (c), and (d) in terms of the tradeoff of compressibility vs. retransmission. What can you conclude?

17.3 *Playback buffer with random arrival time* ⋆⋆

We will look at a question similar to the example in Section 17.3.2 examining the latency–jitter tradeoff, but with a probabilistic packet arrival time. Suppose $V_1 = 0, V_2 = 1$, and $V_3 = 2$, i.e., a step function. Now the packets arrive independently at times A_1, A_2, and A_3, where A_i is drawn randomly between $\bar{A}_i - 1$ and $\bar{A}_i + 1$, and we set $\bar{A}_1 = 3, \bar{A}_2 = 4.2, \bar{A}_3 = 4.6$. What is the optimal playback time of the first packet p^* that minimizes latency but ensures that all packets are received with at least a 95% probability?

17.4 *Round robin, weighted fair queueing, and priority queueing* ⋆⋆

We compare three resource allocation policies. Recall the following.

- Round robin simply gives each queue a turn to transmit a packet.
- Priority queueing allows the queue with the higher priority to be continuously serviced until it is empty.
- A particular implementation of weighted fair queueing looks at the head of each queue and transmits the packet that would finish transmission quickest under the **Generalized Processor Sharing** (GPS) scheme. GPS is an ideal "fluid-flow" scheduler, and is defined as follows: if we have n queues with priority $p_1, p_2, \ldots, p_n$, then the bandwidth allocated to queue j per timeslot is $p_j / \sum_i p_i$.

Suppose we have queue A and queue B with packets arriving:

Time (s)	Queue A arrived packet size	Queue B arrived packet size
$t = 0$		3
$t = 1$	1	
$t = 2$	1	
$t = 3$		2
$t = 4$		
$t = 5$		4

Queue A has priority 1 and Queue B has priority 3 (higher number indicating higher priority). The outgoing link has bandwidth 1 Mbps. Once a packet has begun transmitting, it cannot be pre-empted by other packets. Fill in the following table for round-robin scheduling, priority queueing, and weighted fair queueing.

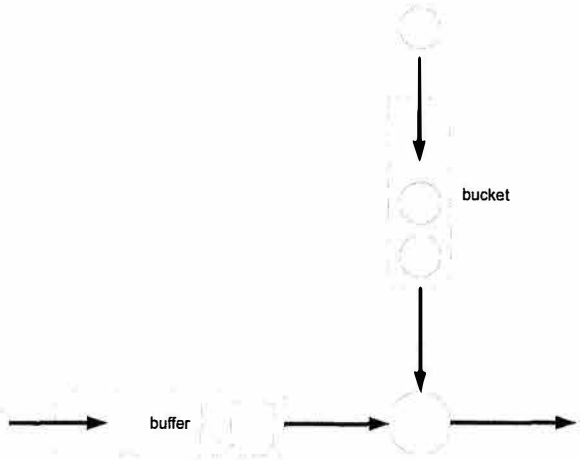


Figure 17.11 An illustration of a leaky bucket for admission control.

Time (s)	Queue A departed packet size	Queue B departed packet size
$t = 0$		
$t = 1$		
$t = 2$		
$t = 3$		
$t = 4$		
$t = 5$		

17.5 *Leaky bucket and GPS* ⋆⋆⋆

A link becomes congested when packets arrive at a faster rate than the link can support. The leaky bucket queueing system is one way to solve this problem. There is a bucket that contains tokens. For traffic class k, the bucket has size B_k and tokens refill the bucket at a rate a_k. Packets wait in the queue and can be released only with a token from the bucket. Therefore, the maximum number of packets that leave the queue during time interval $[u, t]$ is $B_k + a_k(t - u)$. This is illustrated in Figure 17.11.

Several leaky buckets drain into the same buffer. This buffer follows the Generalized Processor Sharing (GPS) service. Eeach traffic class k has weight w_k; C bps is the rate supported by the link out of the buffer; and $\rho_k = \frac{w_k}{\sum_j w_j} C$ is the instantaneous rate at which packets of traffic class k leave the GPS buffer. This is illustrated in Figure 17.12.

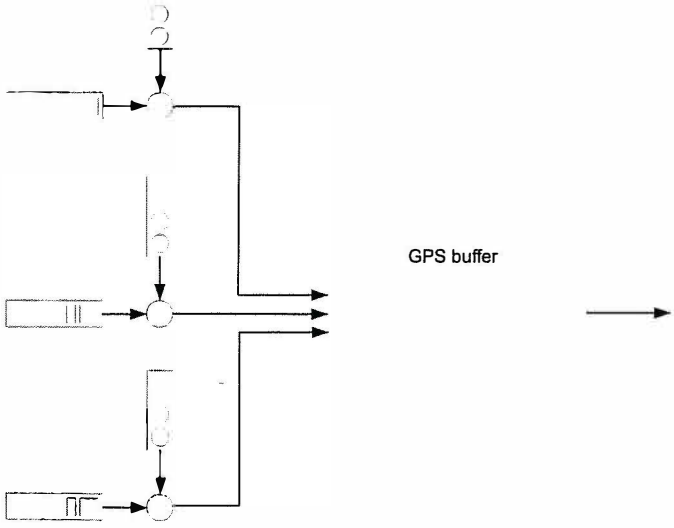


Figure 17.12 An illustration of Generalized Processor Sharing.

(a) During the time interval $[u, t]$, at least $\rho_k(t-u)$ packets leave the GPS buffer. Let $B_{t,k}$ be the backlog of traffic class k in the GPS buffer at time t. Prove that the backlog of class-k traffic in the buffer cannot exceed some B_k if $\rho_k \geq a_k$. To start, assume that there is a time t when the backlog $B_{t,k} \geq B_k$. Consider the largest time $u < t$ such that $B_{u,k} = 0$, and write the inequality relating the change in backlog size from time u to time t.

(b) Prove that the delay experienced by a packet in class k in Figure 17.12 cannot exceed $\frac{B_k}{\rho_k}$ if $\rho_k \geq a_k$.

Let F_k denote the transmission time of packet k (the time when packet k leaves the buffer) under WFQ. Similarly define G_k for GPS. Let L_k denote the size (in bits) of packet k, and L_{max} is the largest L_k. We will show that

$$F_k \leq G_k + \frac{L_{max}}{C}, \ \forall k. \tag{17.1}$$

GPS and WFQ both process packets at the same rate, so the total amount of packets in the system remains the same. Therefore, their busy and idle periods are the same, and we need only show that the result holds for a single busy period. Assume $F_1 < F_2 < \ldots < F_K$ correspond to K packets in one busy period of WFQ.

(c) Pick any $k \in \{1, 2, \ldots, K\}$ and find the maximum $m < k$ such that $G_m > G_k$. (If there is no such m, let $m = 0$.) This implies that $G_n \leq G_k < G_m$ for n in the set of indices $P = \{m+1, m+2, \ldots, k-1\}$. Let T_m denote the time

when WFQ chose to transmit packet m. Now consider the time $S_m = F_m - T_m$. Show that the packets in set P must have arrived after S_m.

(d) Now consider the time interval $[S_m, G_k]$. Packets in set P arrived and departed during this interval. In addition, packet k was transmitted during this interval. Recall that the system is work conserving. Write an inequality relating the $[S_m, G_k]$ to the transmission times of packets in set P, and use this to show the main result in (17.1).

(e) Suppose that multiple-leaky bucket queues are multiplexed to a single WFQ buffer, similar to Figure 17.12. Combine (b) and (d) to show that the maximum delay experienced by a packet of class k is $\frac{B_k}{\rho_k} + \frac{L_{max}}{C}$.